Deputy Director
for Administration

DDA 86-1090
9 June 1986

NOTE FOR: Special Assistant for Development
and Engineering/OIT

SUBJECT: Software

Bill:

I am interested in your comments about the attached article. Note that it states that the development of software is a very slow process compared to the development of hardware. I think we would both accept this statement. How do we improve software productivity?

[redacted] STAT

William F. Donnelly

Attachment

ORIG:DDA:WFDonnelly:be
Distribution:
0 - Adse w/att.
1 - DDA Subj w/att.
1 - DDA Chrono w/o att.
1 - WFD Chrono w/att.

5-1

A GROWING GAP IN SOFTWARE

Computers are getting more and more powerful, but the programs that run them cannot keep up. Business and the Pentagon are taking aim at the problem. ■ *by John Paul Newport Jr.*

THE COMPUTER REVOLUTION is at a turning point. Breakthrough after breakthrough in physics, semiconductor materials, and electrical engineering has created computers that process information at ever higher speeds and ever lower cost. Since 1970 the processing power of computers per dollar spent on them has risen a remarkable 30% a year, on average. But computers cannot run without software, the intricate instructions that tell them how to perform their miracles. And software has not kept up with the stunning improvements in hardware. Software productivity, measured by how quickly developers can churn out useful programs, has increased at only an estimated 4% to 7% annual pace for the past 20 years. The result: a software gap that is the principal barrier to computer progress.

Signs of software trouble are everywhere. In a Wall Street nightmare last November, a glitch in the Bank of New York's software caused almost $25 billion in government bonds it had bought to pile up in its account overnight instead of being sent on to customers; the bank had to borrow most of that amount from the Federal Reserve to pay for the bonds, at a cost of about $5 million in interest. In February ITT reluctantly abandoned U.S. marketing plans for its most important new product, the System 12 telecommunications system, primarily because of software delays. The average backlog of requests for new data-processing programs at 125 companies surveyed in 1984 was 27 months, vs. 19 months in 1981, according to Applied Computer Research, a Phoenix research and publishing firm. Another study found that 75% of the time businesses never use the software programs they undertake, either because they never complete them or because they arrive too late.

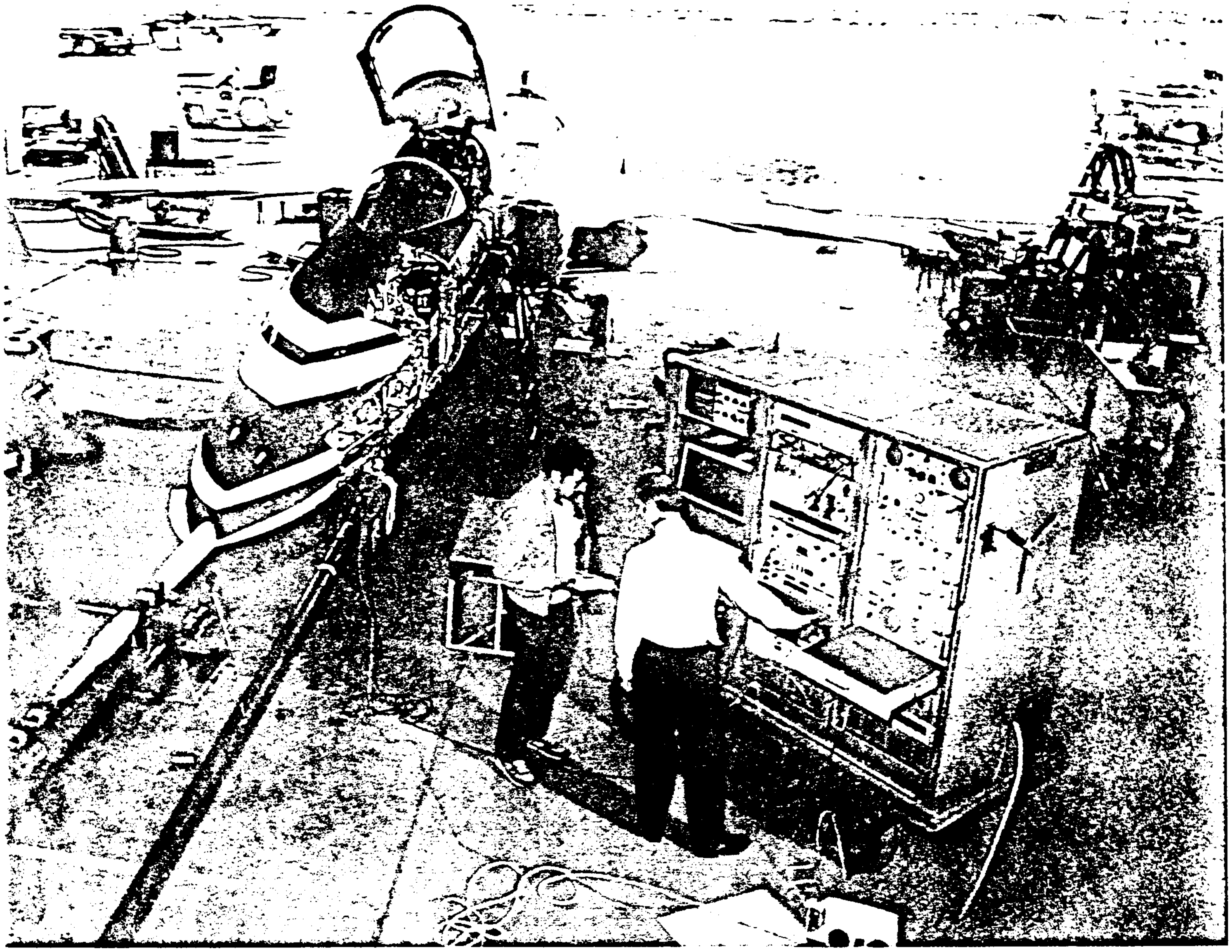
The situation is critical in the military, where software problems are not just a matter of lost opportunity but a potential danger to national security. "We are definitely facing a software crisis," says Edward Lieblein, the Defense Department's director of computer software and systems. Software is crucial to the success of the U.S. commitment to a high-technology defense against everything from Soviet tanks to ICBMs. According to the Institute for Defense Analyses, a private Washington think tank, 80% of the U.S. weapons systems in development depend significantly on software.

The Pentagon and its contractors are already having trouble fielding weapons with relatively simple software. A major shortcoming of the abandoned Sergeant York antiaircraft gun was that contractors could not write tracking software sophisticated enough for it to hit aircraft making evasive maneuvers. The Department of Defense recently certified that the long-delayed Advanced Medium-Range Air-to-Air Missile (AMRAAM) meets its performance specifications. But the missile has not been fully tested in the field and congressional critics threaten to kill it, at least in part because the prime contractor, Hughes Aircraft, has not perfected the missile's guidance software.

Software is also the biggest stumbling block to President Reagan's "Star Wars" antimissile system, officially called the Strategic Defense Initiative (SDI). Programming to control the dispersed weapons, satellites, and command centers would constitute the largest and most complex software project ever attempted. It could require anywhere from ten million to 35 million intricately interrelated instructions to computers. A word-processing program for a personal computer has about 5,000 instructions, the Lotus 1-2-3 spreadsheet some 100,000, and the operating system for a typical large computer about 500,000. Few software systems have more than a million instructions. Last

RESEARCH ASSOCIATES *Joan Campo and Brett Duval Fromson*

Without intricate software, *the experimental*

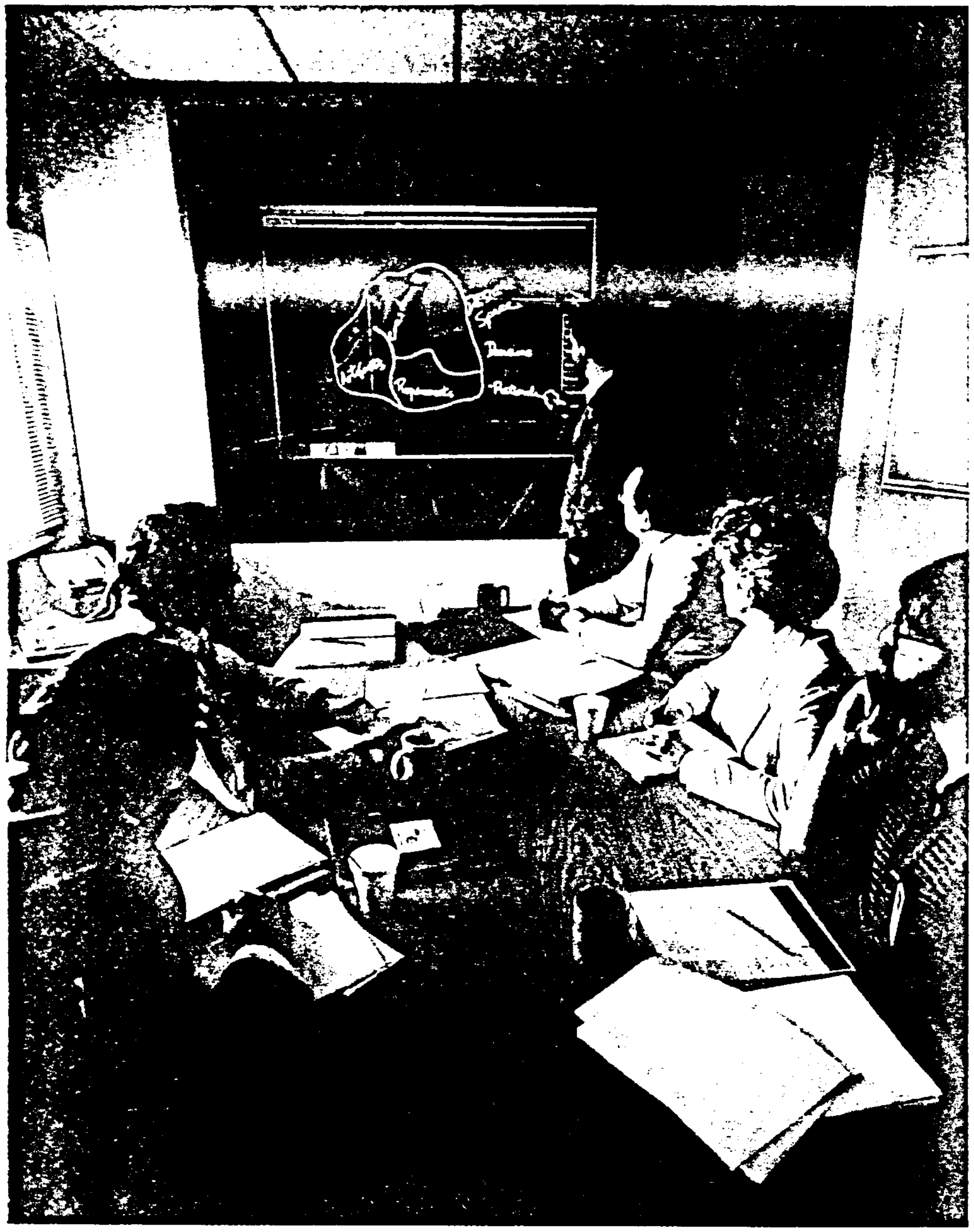


undergoing tests at California's Edwards Air Force Base, could not fly. It would break up if computers failed to adjust the controls 40 times a second.

summer David Parnas, a prominent computer scientist who analyzes complex programming problems, resigned from an SDI software advisory panel after concluding that reliable software for Star Wars could never be built. Most other experts working on SDI believe reliable software can be developed with enough time and research money, but only if planners make software the paramount technical priority. That would mean an about-face for the Pentagon, whose usual practice is to develop weapons first and hope software can be written for them later.

The Pentagon spent about $11 billion last year on so-called mission-critical software for weapons systems, communications, and intelligence gathering. The Electronic Industries Association projects that by 1990 the amount will more than double, accounting for roughly 20% of everything the Pentagon spends on weapons. Unless productivity increases dramatically, the association says, by then the U.S. could need a million more programmers than will be available to handle the job. The solution is more productivity, not more programmers. Star Wars may seem far removed from the run-of-the-mill software problems confronting business, but the basic goal is the same: to increase productivity by finding ways to relieve software developers of time-consuming drudgery, so they can concentrate on tough conceptual problems.

An important first step is giving software developers computers to work with. Like the cobbler's children who have no shoes, most software designers still work with paper and pencil because little satisfactory software has been designed for them. Automation is the foundation for almost everything researchers have in the wings to improve software productivity. Computers will give programmers instant access to the work of



Researchers *at Microelectronics & Computer Technology Corp. in Austin, Texas, write on an electronic chalkboard, part of a computer network they use for studying software productivity.*

colleagues that relates to their own, for example, and help managers collect performance data and monitor the progress of a project—things they find difficult if not impossible to do today. A number of companies are beginning to offer packages that help automate software design (see box). The aim of the industry is to transform software from art to engineering, from a handicraft that commonly relies on intuition and trial and error to a discipline with a body of basic principles, standardized practices, and rigorous management control.

The challenges loom largest for the Pentagon because of the inherent complexity of military software. One industry newsletter reported that an early version of the F-16 fighter's navigation software would have caused the plane to flip over anytime it crossed the equator. That flaw was caught during computer simulation testing, but many less dramatic mistakes are not.

Weapons must often be used in unpredictable or unanticipated circumstances. Unlike business data, which programmers feed into computers in organized, neatly typed fashion, computer input for controlling weapons must come from sensors like radar or electrooptical devices. Sensor data pour forth as millions of raw numbers per second. The computer software must analyze the data and decide whether they describe, say, an enemy tank or a friendly Volkswagen. To be useful, the software must deliver its verdict in real time—that is, almost instantaneously. Making matters tougher, the nose cone of a missile or the electronics bay of a fighter provides limited space and electrical power for data-processing equipment. The software must use the fewest possible steps to do its job, which makes it harder to design.

MILITARY SYSTEMS also need extra-reliability. Sometimes designers create several different programs that simultaneously perform the same function, such as controlling an airplane's automatic pilot. If one program fails, the others take over. Developers used this technique in designing part of the software for the experimental X-29 fighter, whose forward-swept wing design is so aerodynamically unstable that three on-board computers must adjust the plane's pitch and roll 40 times a second. If all three computers fail, the plane would break apart in midair.

Defense Department analysts recognized a decade ago that the proliferation of computers could create a software crunch, but not until 1984 did the Pentagon require the use of Ada, an advanced new computer language, for all new weapons systems. In the same year it established the Software Engineering Institute at Carnegie-Mellon University to identify and apply promising new research and technology that could be useful in everyday software development. A government-funded venture called STARS (for Software Technology for Adaptable, Reliable Systems) will spend $250 million over the next five years to coordinate Pentagon software policy and sponsor software research.

Defense contractors share the Pentagon's concern. Last year 14 of the biggest, including General Dynamics, Lockheed, McDonnell Douglas, and Boeing, committed $2.8 million each to form the Software Productivity Consortium in Reston, Virginia. It will pool research to perfect advanced software development methods that members can use in their work for the Pentagon. Software research is one of the four major projects at another consortium, Microelectronics & Computer Technology Corp. of Austin, Texas. Among other things, the company studies the psychology of teams designing complicated systems. It plans to use what it learns to design programs that help programmers build software.

None of these efforts has yet produced an enchanted sword that alone will slay the software dragon. The three areas of greatest promise for the military are broadening the use of Ada, making software reusable, and developing new techniques for helping computer users specify at the outset exactly what results they want from software.

OF THE THREE, Ada is the most important in the near term. "The power of Ada remains largely undiscovered by the aerospace industry," says Winston Royce, director of Lockheed's software technology center in Austin. Royce says companies can achieve 50% gains in productivity the first time they use the language. Designed to Pentagon specifications in the late 1970s, Ada has features that prevent some types of common programming errors and that make programs written in it easier to understand and modify.

The greatest hope for Ada, however, is in software reusability, something of a holy grail in the industry. Using a program for more than one application is uncommon. One problem is that even programs in the same computer language are often incompatible, and at last count the Pentagon, for example, was using a Babel of 400 different languages, or variations of them, in its weapons systems. Ada makes it easier to reuse programs, partly because programs written in it can be run without change on different kinds of computers. "Obviously if you can reuse software that already exists rather than build a program from scratch, you're miles ahead," says Larry Druffel, former head of the Pentagon's Ada program and now a vice president of Rational, a Mountain View, California, company that makes a software development system that uses Ada.

Hughes Aircraft, a subsidiary of General Motors, was able to reuse 70% of the software it designed for one foreign country's air defense system in the only slightly different air defense system it built for another foreign client. The savings: 50% of the software cost. But most software projects are not so similar. A major barrier to reuse is simply the difficulty of locating previous programs that could be helpful because they addressed the same kind of problem; so far they have not been systematically catalogued. The first project of the Software Productivity Consortium is to design a kind of Dewey Decimal System for software, [illegible] begin to build software libraries.

Nailing down precisely what software systems are supposed to do is the hardest phase of software development to automate, but it has the greatest potential for improving productivity. AT&T's Bell Laboratories estimates that it costs nothing to fix an error in the design phase, $100 to fix the same error once programmers have coded it, and $10,000 to fix it in the field. Often the problem is vague specifications by the person or organization that will be using the system. In the late 1970s the Army paid TRW over $60

AFTER CAD, CAM, AND CAE, NOW COME THE CASE COMPANIES

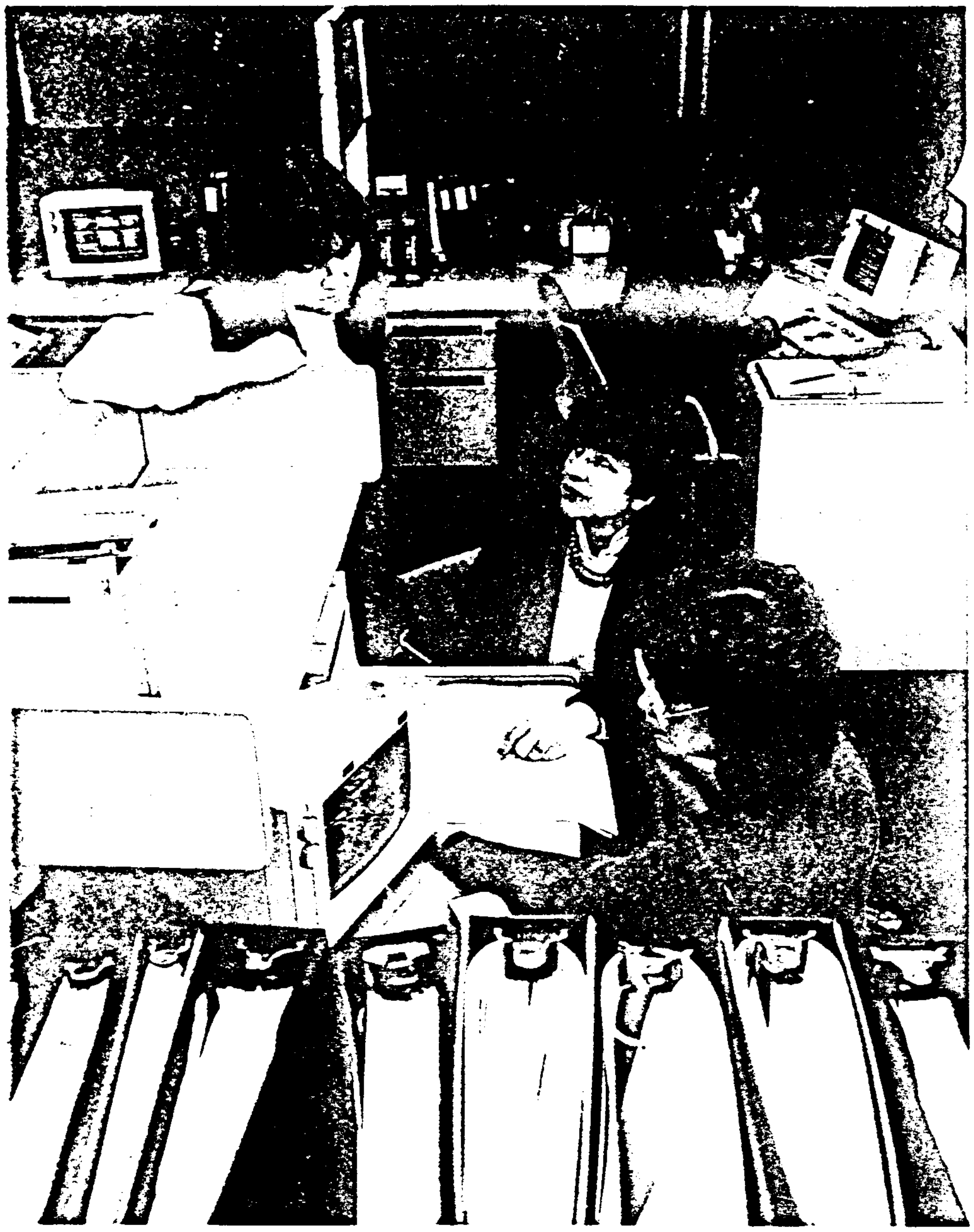
■ First there were the twins CAD and CAM, computer-aided design and computer-aided manufacturing, software systems linked with computer work stations to speed the conception and production of all sorts of goods. Then came CAE, computer-aided engineering, which does the same thing for the design of electronic circuits. Now comes software help for designing software itself. The nascent industry is still struggling to find a name and an acronym, but most people in the business like CASE—computer-aided software engineering. Industry sales last year were less than $20 million, but with about one million programmers working on software development in the U.S. alone, the companies expect rapid growth. One computer marketing research firm estimates the potential market at $1 billion a year by 1990.

CASE systems help in the difficult early stages of software development, where teams of designers reduce large problems to successively smaller parts. "If you make mistakes in the design phase, automating what comes later just means you're building the wrong system three times faster," says Lou Mazzucchelli, chairman of Cadre Technologies, a Providence, Rhode Island, maker of CASE products. Other companies in the business include Index Technology of Cambridge, Massachusetts; Nastec of Southfield, Michigan; KnowledgeWare Inc. of Ann Arbor; and Hitech, a joint venture of the Hartford Insurance Group and Wang Laboratories. All are are privately held.

A typical CASE system is made up of five to ten separate programs, collectively called an environment, that coordinate the complicated work of software design. The system allows engineers to create on computer screens the flow charts they previously drew with pencil and paper, and to fill in the empty boxes and circles in the chart with information from central databases called dictionaries or encyclopedias. The dictionaries also keep track of all of the parts of a program and the relationships among them. That can help enormously in deciphering the program months or years later when changes or updates are required.

James McGuire, *president of Nastec Corp., shows off his company's DesignAid system.*

The biggest advantage of CASE-based programming may be that it encourages teamwork. Software designers work independently but share information through the dictionary. Rules built into the software check for inconsistencies in programming practice, such as forgetting to put all the pieces together in the flow chart. The basic software for most CASE systems costs $5,000 to $10,000; with personal computers or engineering work stations to run the software, the cost per engineer comes to $15,000 or more. CASE companies claim the investment can increase productivity in software development 100% to 300%.



Software developers *like these at the Hartford Insurance Group have boosted programming productivity 27%. The company markets its computer network system with Wang Laboratories.*

million to build a huge communications system called BETA (for Battlefield Exploitation and Target Acquisition). BETA was to gather and assimilate data from many sensors and display the results to help commanders in the field make decisions. "The system did exactly what the Army thought it wanted the system to do," says Larry McLaughlin, a manager of engineering technology at TRW. "The trouble was, no one could use it. It gave them too much information."

To avoid such disasters, TRW and other companies are pioneering a process called rapid prototyping. The idea is to give users the look and feel of a real system before it is built. TRW's prototyper, which contains 750,000 computer instructions, enables customers to play with simulated versions of programs they want TRW to build. Naval officers using the prototyper to develop a ship-tracking and ocean-surveillance system recently found they could get by with a single high-resolution display screen instead of with two.

While the Pentagon and defense contractors such as Lockheed, TRW, and Boeing are pouring millions of dollars into these projects and into still more advanced software research to leapfrog beyond the current state of the art, the civilian world is concentrating on the here and now. Some commercial enterprises have adopted solutions that are more advanced in many respects than military practices. In the late 1970s the Hartford Insurance Group found itself squeezed by programming problems even though the staff and budget for software development had grown 25%. So the company began a crash program to apply the latest programming methods, such as reusing parts of old programs.

THE REDUCTIONS in program backlogs were so encouraging that the company, working with computer maker Wang Laboratories, is now halfway through phase two: installing computer work stations and sophisticated software packages for everyone in its software development department, secretaries and programmers alike. The company says initial productivity gains of 27% should increase with experience and repay the project's $18-million cost in 18 months. Backlogs have declined further, and by 1987 the software development department will have reduced its payroll from more than 1,200 to 1,000.

Financial and telecommunications networks, airline reservation systems, and factory automation must operate at high speed with high reliability. As a result, companies in these areas are doing much software research that mimics what is being done by the military. General Motors, with the help of its acquisition, Electronic Data Systems, is spending millions of dollars to develop software for integrating a multitude of factory automation tasks into a single system, a challenge not unlike that facing builders of large military communications systems. AT&T's Bell Laboratories has 8,000 people in software research, including one team developing a way to catalogue software components for reuse in telecommunications networks.

In the near term, says Barry Boehm, chief engineer of TRW's systems development division, most organizations could double software productivity using nothing more than the best software practices and technologies already available. A pilot project under Boehm's direction at TRW has increased productivity by 40% since 1982. The primary barrier to progress is not so much technology as what Dan Appleton, president of a Los Angeles consulting firm, and others call "the will problem": getting top executives, most of whom rose through the ranks before the software gap developed, to understand the software challenge and commit themselves to meeting it. ◘